Introduction au Javascript

Du back-end au front-end, une introduction au JS via Python

Thibault Clérice,
École Nationale des Chartes
https://github.com/ponteineptique/cours-javascript

Objectifs du cours

- Cours 1
  - Comprendre ce qu’est le javascript
  - Le javascript dans le HTML
  - Comment déboguer du javascript
  - Syntaxe du javascript (par rapport à python)
  - Le DOM
- Cours 2
  - L'usage de bibliothèques
  - Le concept d'Ajax
  - Le concept de fonctions anonyme et de *callback*
  - Utilisation d'une librairie
    - Faire un pop-up à partir de données Ajax
- Cours 3 et suivants
  - Faire une carte avec Leaflet
  - Ajout de champs en HTML

Bibliographie

- Php, MySQL et Javascript, Nixon, O’Reilly
- JavaScript & JQuery, Jon Duckett, Wiley & Sons
- Eloquent Javascript, Haverbeke, O’Reilly (3Eme édition : http://eloquentjavascript.net/3rd_edition/)

Bibliographie (2)

- Cours
  - http://fr.eloquentjavascript.net
  - http://www.gchagnon.fr/cours/dhtml/
- Outils
  - https://jsfiddle.net/
- Veille
  - https://reddit.com/r/javascript
  - http://stackoverflow.com

Avertissement

Ce cours n'a pas été écrit en utilisant la syntaxe nouvelle de ES 6 et ES 7, qui sont les dernières versions de javascript. Il est possible que les manuels les plus à jour utilisent cette nouvelle syntaxe, très proche de celle utilisée ici.

Qu'est-ce que javascript

Courte histoire du javascript

Où se situe le javascript dans le Web ?

Le javascript dans du HTML : en insertion directe

```
<html>
  <head>
    <script type=”text/javascript”>
      // Le javascript ici
    </script>
  </head>
  <body>
    <script type=”text/javascript”>
      // Le javascript ici
    </script>
  </body>
</html>
```

Le javascript dans du HTML : en insertion externe

```
<html>
  <head>
    <script type=”text/javascript”
      src=”./chemin/vers/javascript.js”></script>
  </head>
  <body>
    <script type=”text/javascript”
      src=”./chemin/vers/javascript.js”></script>
  </body>
</html>
```

Déboguer du javascript

- Firefox
  - Outils > Web Developer > Debugger
  - Ctrl + Maj + S // Pomme + Maj + S
- Chrome
  - Ctrl + Maj + I // Pomme + Maj + I

Développer du javascript

- IDE : WebStorm (PyCharm de Javascript)
- Éditeur de texte avancé:
  - Sublime
  - Atom
  - Bracket

Syntaxe du javascript (1)

Général

1. Une ligne se finit par un `;` quand une instruction est finie.
2. Les variables se définissent comme en python :

```
mon_nombre = 456;
ma_chaine = "Hello !";
```

4. Il est de bonne pratique de précéder la variable par `var` quand on la crée.
5. On commente avec `//` sur une seule ligne ou avec `/*` et `*/` sur plusieurs :

```
var mon_nombre = 456;  // N'est-ce pas magnifique ?
/* Je peux écrire
un commentaire sur plusieurs lignes
*/
var ma_chaine = "789";
```

Syntaxe du javascript (2)

Blocs conditionnels

1. Les blocs `if-elif-else` s'écrivent avec des accolades `{}`.
2. On entoure les conditions avec des parenthèses `(variable == True)`
3. `and` pour que deux conditions soient valides s'écrit `&&`
4. `or` pour qu'une condition soit valide sur les deux s'écrit `||`

```
ma_variable = 123;
if (ma_variable == 123) {
        // Que faire ?
} else if (ma_variable == "123") {
        // Que faire ?
} else {
        // Que faire ?
}
```

Syntaxe du javascript (3)

Égalités

Les égalités s'expriment de la même manière qu'en Python. On fera attention cependant aux variantes `===` et `!==` qui signifient strictement égales ou inégales. On les préfèrera à `==` et `!=`. *Cf.* https://dorey.github.io/JavaScript-Equality-Table/

Opérateur	Exemples qui renvoient true
Égalité (==)	`3 == var1` `"3" == var1` `3 == '3'`
Inégalité (!=)	`var1 != 4` `var2 != "3"`
Égalité stricte (===)	`3 === var1`
Inégalité stricte (!==)	`var1 !== "3"` `3 !== '3'`
Supériorité stricte (>)	`var2 > var1` `"12" > 2`
Supériorité ou égalité (>=)	`var2 >= var1` `var1 >= 3`
Infériorité stricte (<)	`var1 < var2` `"2" < "12"`
Infériorité ou égalité (<=)	`var1 <= var2` `var2 <= 5`

Syntaxe du javascript (4)

Dictionnaires et listes

1. Les dictionnaires et listes sont appelés `Object` et `Array`.
2. Les dictionnaires sont en fait des objets (au sens instance de classe avec `self` en python) simplifiés
3. L'écriture d'une liste ou d'un dictionnaire sont similaires :

```
var dico = {
  "python": "Un langage propre",
  "javascript": "Un langage discuté",
  "php": "Personne ne m'aime",
  "delphi": "?"
}
var liste = ["a", 1, "4", dico]
```

4. L'accès se fait comme en python : `liste[0] === "a"` et `dico["delphi"] === "?"`. Mais on peut ajouter l'écriture `dico.delphi === dico["delphi"]` comme écriture.

Syntaxe du javascript (5)

Dictionnaires et listes

Description	Python	Javascript
Assignation	`x= [1, 2, 3]`	`var x = [1, 2, 3]`
Taille	`len(x) == 3`	`x.length === 3`
Vide	`not x`	`x.length === 0`
Element `i`	`x[i]`	`x[i]`
Dernier élément	`x[-1] == 3`	`x[x.length-1] === 3`
Sous-ensemble	`x[1:2] === [2, 3]`	`x.slice(1,2) === [2, 3]`
Sous-ensemble jusqu'à la fin	`x[1:] == [2, 3]`	`x.slice(1) === [2, 3]`
Ajout d'un élément	`x.append(4)`	`x.push(4)`
Tri	`x.sort()`	`x.sort()`
Inclut	`2 in x`	`x.includes(2)`
Additon de listes	`[1] + [2] == [1, 2]`	`[1].extend([2]) === [1, 2]`

Syntaxe du javascript (6)

Les boucles

```javascript
// Javascript
var liste = [6, 7, 8];
for (var entier of liste) {
  console.log(entier * 2);
}

// Énumeration basée sur une liste
for (var i = 0; i < liste.length; ++i) {
  console.log(liste[i]);
}

// Énumeration basée sur une range
for (var i = 0; i < 100; ++i) {
  console.log(i);
}

// Pas avec while
var i = 0;
while (i < 100) {
  console.log(i);
  i += 1;
}
```

```python
# PYTHON
liste = [6, 7, 8];
for entier in liste:
  print(entier * 2)


# Énumeration basée sur une liste
for index, entier in enumerate(liste):
  print(entier == liste[index])


# Énumeration basée sur une range
for i in range(0, 100):
  print(i)


# Pas avec while
i = 0;
while i < 100:
  print(i)
  i += 1
```

Fonctions

```javascript
/**
 * Conjugue un verbe du 1er groupe
 *   au présent
 *
 * @param {str} v Verbe à l'infinitif
 * @param {int} p Personne
 * @param {int} n Nombre
 *     (1 si singulier, autre si pluriel)
 * @return {str} Verbe conjugué
 */
var conjugue = function(v, p, n) {
 var verbe = v.substring(0, v.length-2);
 if (n === 1) {
   if (p === 1) {
     return verbe + "e";
   } else if (p === 2) {
     return verbe + "es";
   } else if (p === 3) {
     return verbe + "e";
   }
 } else {
   if (p === 1) {
     return verbe + "ons";
   } else if (p === 2) {
     return verbe + "ez";
   } else if (p === 3) {
     return verbe + "ent";
   }
 }
}

console.log(conjugue("chanter", 2, 2));
```

```python
def conjugue(verbe, personne, nombre):
 """ Conjugue un verbe du 1er groupe
 au présent

 :param verbe: Verbe à l'infinitif
 :type verbe: str
 :param personne: Personne
 :type nombre: int
 :param nombre: Nombre
      (1 si singulier, autre si pluriel)
 :type nombre: int
 :returns: Verbe conjugué
 :rtype: str
 """
 verbe = verbe[:-2]
 if nombre == 1:
   if personne == 1:
     return verbe + "e"
   elif personne == 2:
     return verbe + "es"
   elif personne == 3:
     return verbe + "e"
 else:
     return verbe + "ons"
   elif personne == 2:
     return verbe + "ez"
   elif personne == 3:
     return verbe + "ent"


print(conjugue("chanter", 2, 2)
```

Attention : pas de paramètres nommés en javascript ES5. On utilisera un dictionnaire si on en ressent le besoin.

Exercice

Le DOM